MAPL3:
A Hierarchical Framework Based on ESMF

T. Clune[1]

Atanas Trayanov[1,2] Arlindo da Silva[1],

Ben Auer[1,2] Hamid Oloso[1,2]

Gerhard Theurich[3,4]

Matt Thompson[1,2]

[1] NASA GSFC
[2] SSAI
[3] NRL
[4] SAIC

What is MAPL

MAPL: a *usability layer* on top of ESMF to support the GEOS ecosystem of Earth system models

➤ Designed to assemble ESMF components in a ***hierarchy***

➤ Co-designed with ESMF from the beginning

➤ Lightweight specification of import, export, and adding child components.

➤ Services:
  ➤ Automated allocation of fields
  ➤ Default implementation for checkpoint/restart.
  ➤ Generalized output - History GridComp
  ➤ Generalized input – ExtData GridComp
  ➤ Universal CapGridComp

➤ External users: Harvard GCHP, NOAA UFS

Similar motivations led to the later development NUOPC but with significant divergence in important details.   Interoperability is a nontrivial challenge.
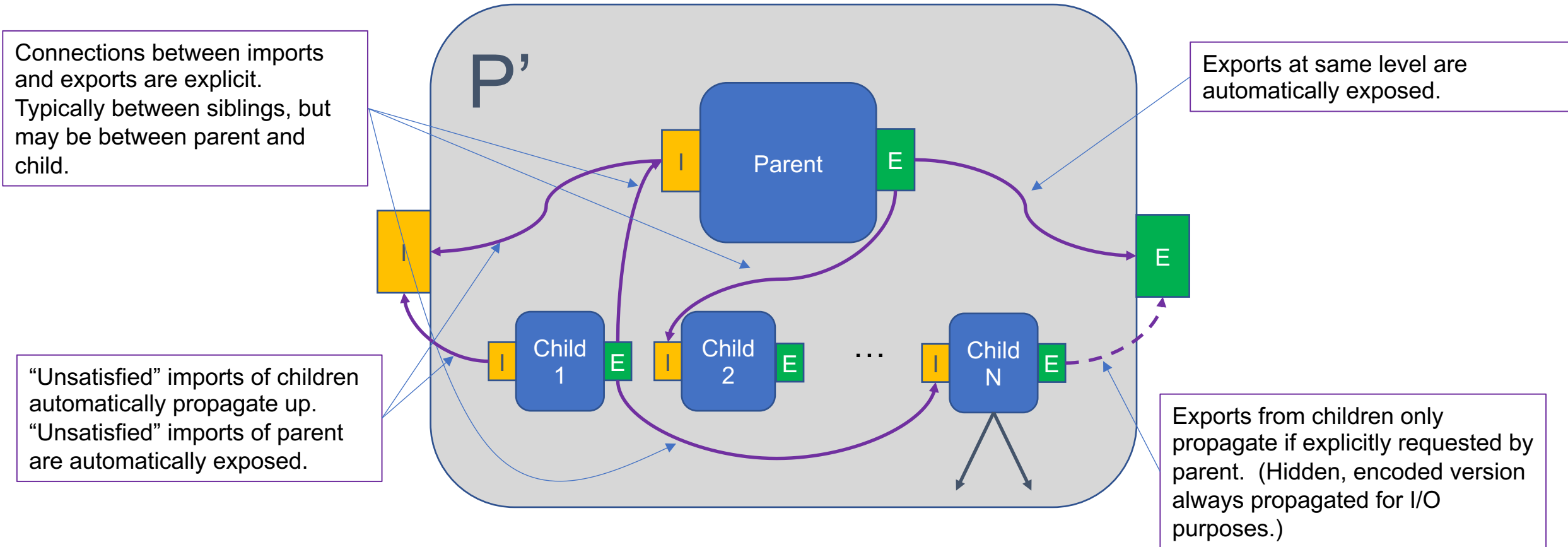

https://github.com/GEOS-ESM

https://github.com/GEOS-ESM/MAPL
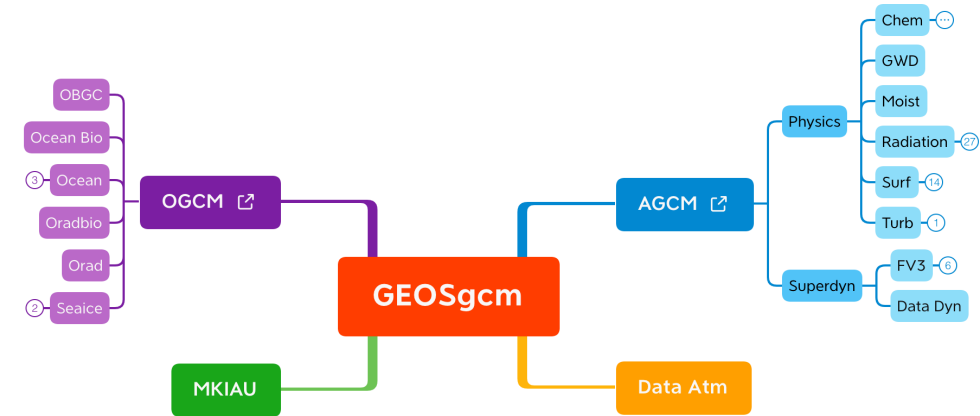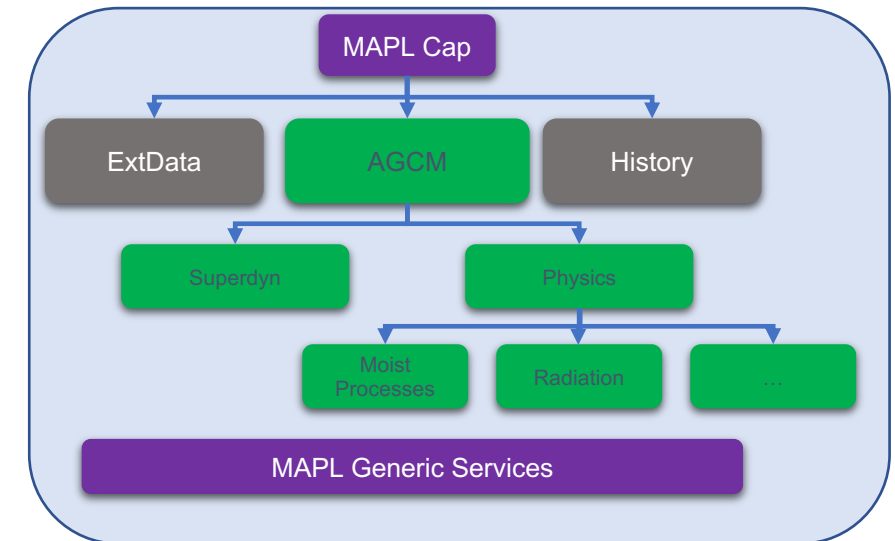
Key differences between NUOPC and MAPL3

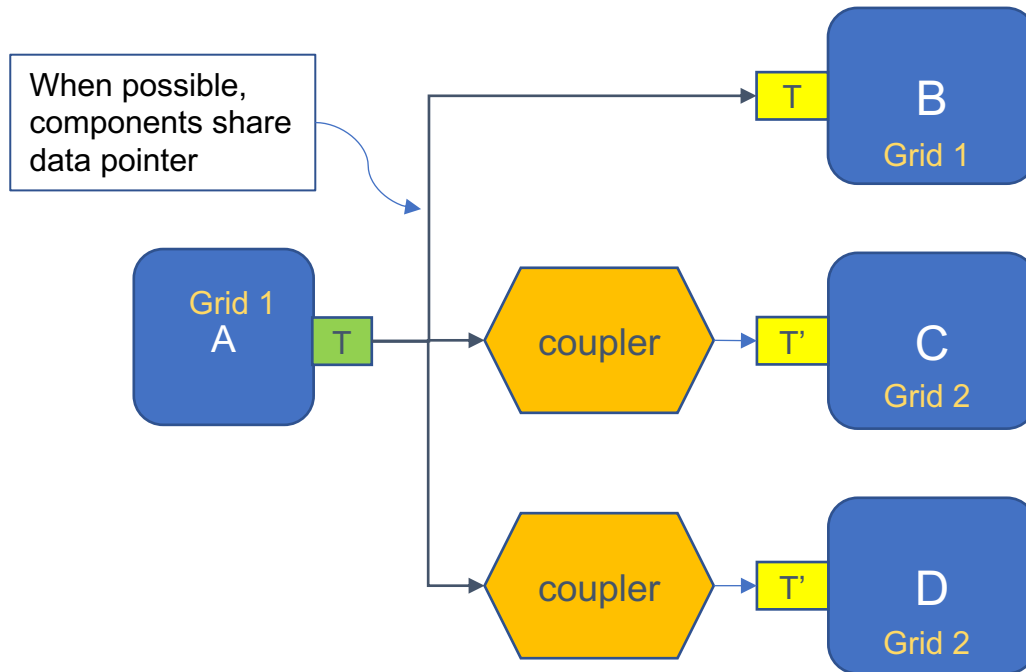| Feature	MAPL	NUOPC
Community support	✗	✓
Concurrent execution of components	✗	✓
Parameterized run sequences	✗	✓
Hierarchical system of components	✓	😑 (partial)
Automatic connections (based on standard names)	✗	✓
Automatic coupling	extensions	couplers
Regrid	✓	✓
Device copy (GPU)	✓	✗
Grid inheritance	✓	🤝 👉 👉 (per field)
MAPL services		
Service services	✓	✗
Automated support for checkpoint/restart	✓	✗
Automatic connections	✗	✓ (standard names)
Automated field allocation	✓	✗

Impetus for continued development

- ➢ Performance/scalability
  - ➢ Support for efficient use of GPUs
  - ➢ Hybrid MPI + OpenMP
  - ➢ Run child components on coarser grid

- ➢ NUOPC interoperability
  - ➢ External collaborations
  - ➢ Concurrent execution (e.g., Ocean-Atmos)

- ➢ Recognition that several existing "cheats" are fragile
  - ➢ Mostly unwarranted assumptions about pointer sharing for data
  - ➢ Parents directly accessing data through import/export state of child
  - ➢ Components modifying data in *import* states
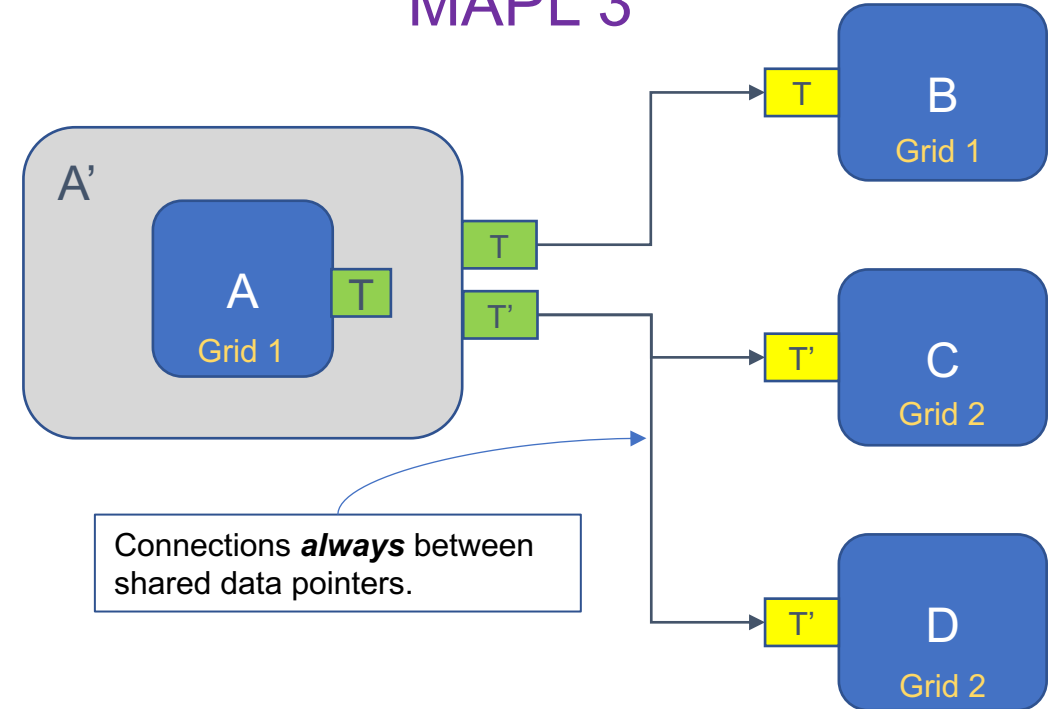
Formal Treatment of Hierarchy

Connections between imports and exports are explicit. Typically between siblings, but may be between parent and child.

Exports at same level are automatically exposed.

"Unsatisfied" imports of children automatically propagate up. "Unsatisfied" imports of parent are automatically exposed.

Exports from children only propagate if explicitly requested by parent. (Hidden, encoded version always propagated for I/O purposes.)

P'

Parent

I   E

I   E   I   E   ...   I   E

Child 1   Child 2   Child N

I   E

Extensions vs Couplers

MAPL 2

When possible, components share data pointer

Grid 1 A — T → coupler → T' C Grid 2

T B Grid 1

coupler → T' D Grid 2

Couplers are between pairs of components. This can result in duplicated work and storage when multiple imports require the same data transformations.

MAPL 3

A'
A Grid 1 — T
T
T'

T B Grid 1

T' C Grid 2

T' D Grid 2

Connections **always** between shared data pointers.

T' is an "extension" that is automatically computed by MAPL.

MAPL automatically constructs the extended component A' and executes "actions" to generate extensions as needed.

Virtual & Actual Conneciton Points (CPs)

User GridComps express all imports, exports and connections in terms of *virtual* CPs

Each virtual CP corresponds to a list of *actual* CPs. Encoded to prevent name collisions.
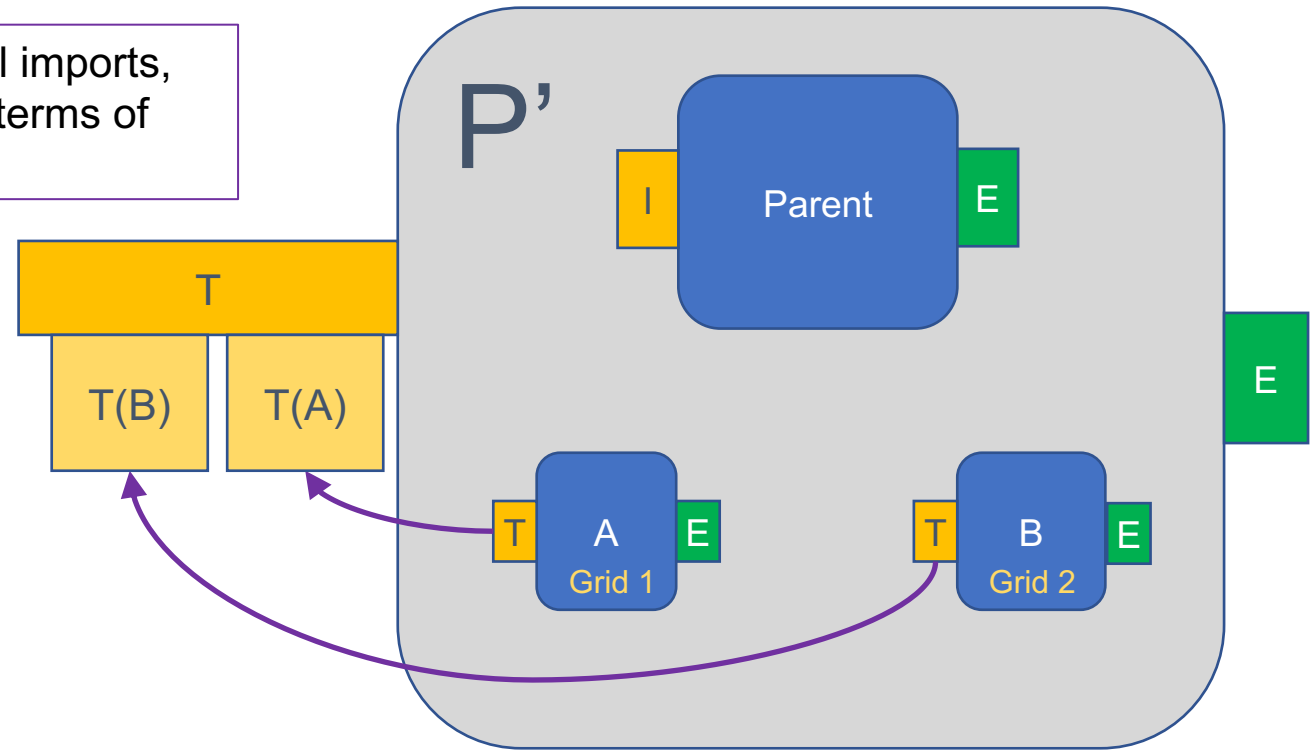
Virtual and Actual CPs (cont'd)

A **virtual** connection between two virtual CPs must satisfy **all** associated actual import CPs
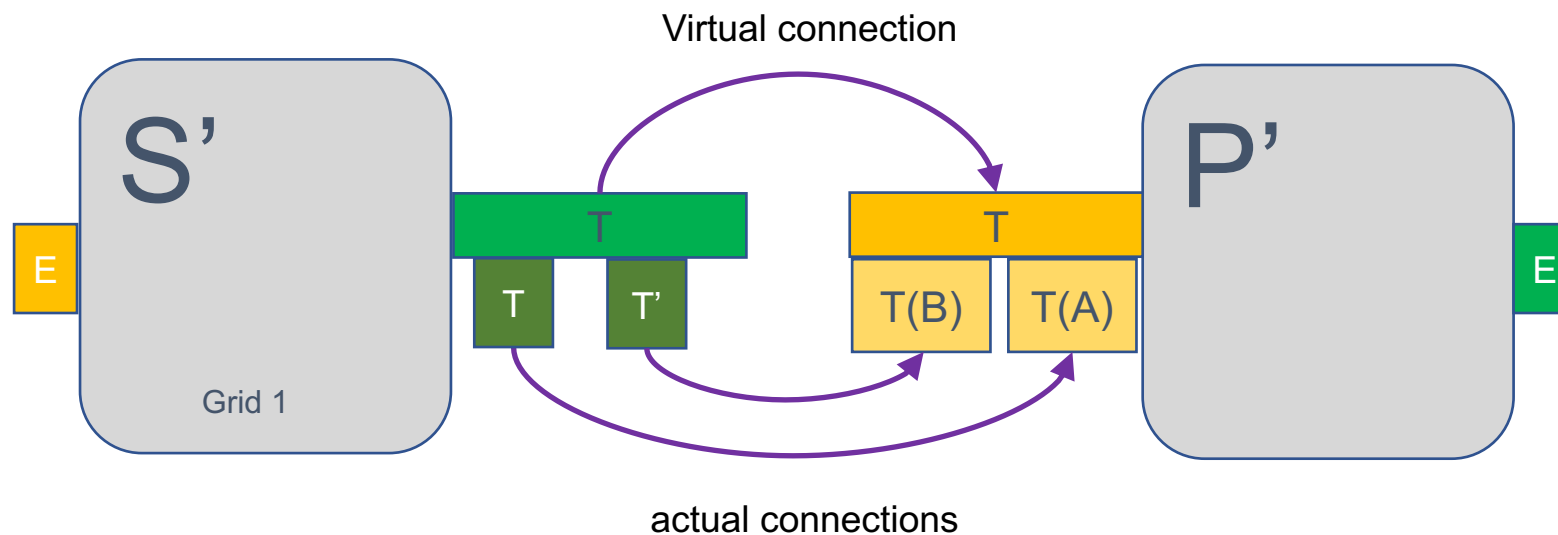
When no current actual exports matches, MAPL extends component with actions:

      Regrid (horz, vert)
      Time accumulation/averaging
      Unit change
      Device copy

Virtual connection

S'

P'

E

T

T   T'

T

T(B)   T(A)

E

Grid 1

actual connections

Generalization of Component Data Services

MAPL 2 categories of import/export items

➢ ESMF_Field

➢ ESMF_Bundle

➢ ESMF_State

MAPL 3 introduces the following additional categories:

➢ Service Services

    ➢ Allow parent component to control which grid comp provides service to children:

        ➢ Advection

        ➢ Turb convection, etc.

    ➢ Prototyped in MAPL 2

➢ Tangent Vector:

    ➢ (u,v) do not regrid as a pair of scalars

These extensions are implemented under-the-hood with proper ESMF data objects.
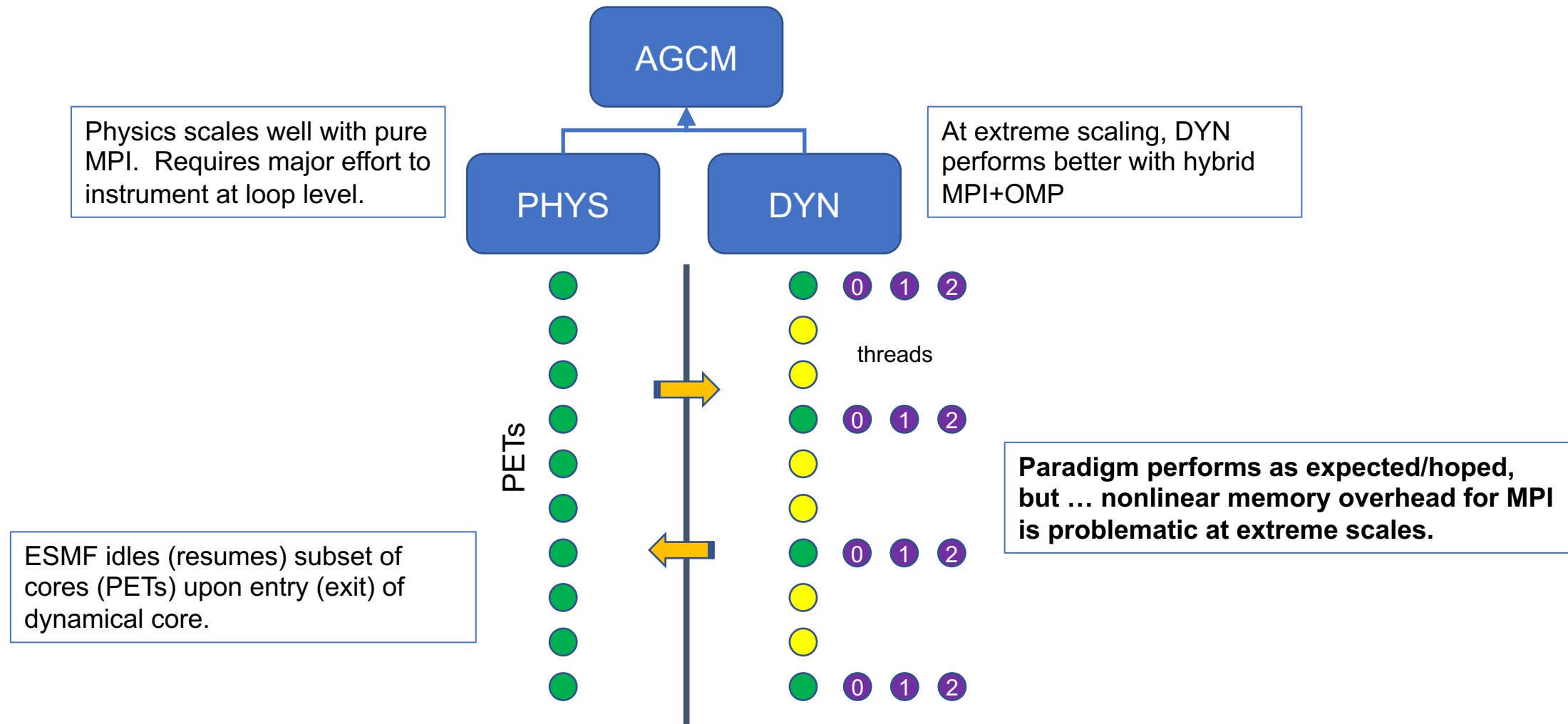
Exploiting GPUs

By default, a component that executes on a kernel must copy *all* imports from the host to the device and *all* exports from the device to the GPU.
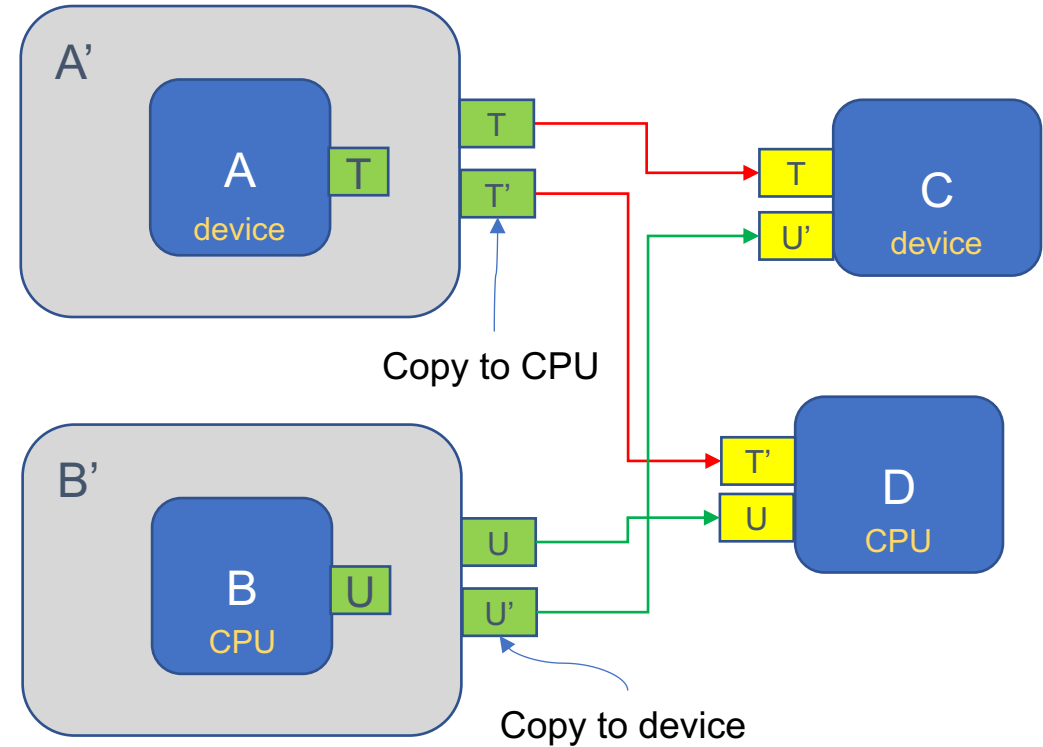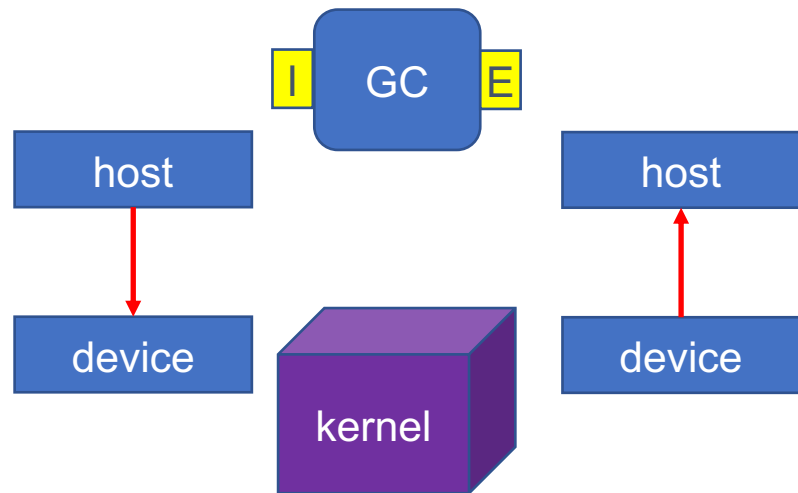
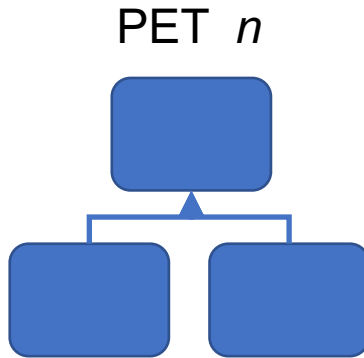In addition to regridding, MAPL extensions can copy to/from device.

Components must register which phases run on device/cpu, And *should* specify which imports/exports are involved in each phase. This allows framework to minimize number of copies.
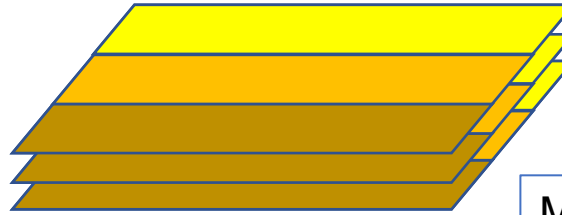
Hybrid OpenMP – Prior approach

Physics scales well with pure MPI. Requires major effort to instrument at loop level.

At extreme scaling, DYN performs better with hybrid MPI+OMP

Paradigm performs as expected/hoped, but … nonlinear memory overhead for MPI is problematic at extreme scales.

ESMF idles (resumes) subset of cores (PETs) upon entry (exit) of dynamical core.

Hybrid OpenMP: new approach

PET *n*

subdomain *n*

MAPL subtree to be threaded
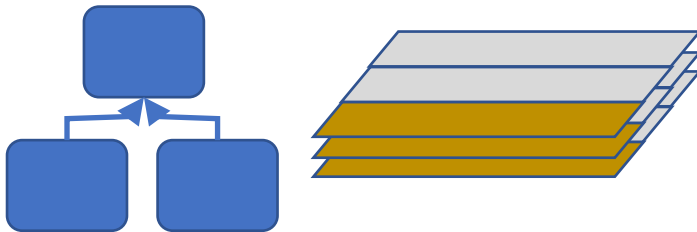
Primary ESMF decomposition - pure MPI running on subset of cores.

MAPL builds suite of per-thread "mini" subtree composed of "mini" gridcomps and "mini" states.
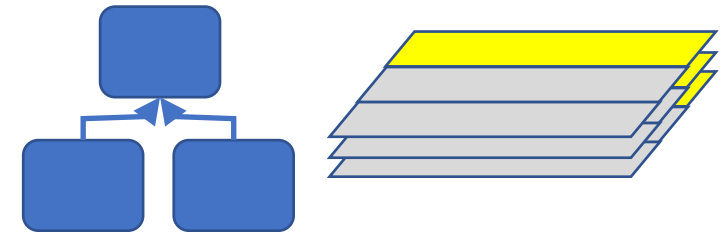
Thread 0

Thread 1

Thread 2

Data pointers in mini-states are associated with **slices** of the primary ESMF decomposition.
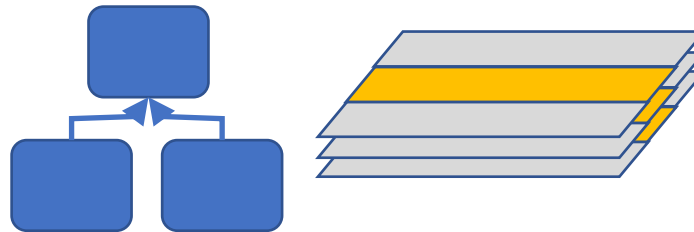
Rest of hierarchy must either run in serial or explicitly instrument with OpenMP.

NUOPC Interoperability

➤ Weak interoperability: MAPL component subtree can be used as a NUOPC model

➤ Strong interoperability: Weak + NUOPC model can be used as a MAPL component

Changes in MAPL

- Introduction of additional init phases
- Major re-engineering of field allocation algorithm
- Provide mechanism for driver to specify grid of top MAPL component
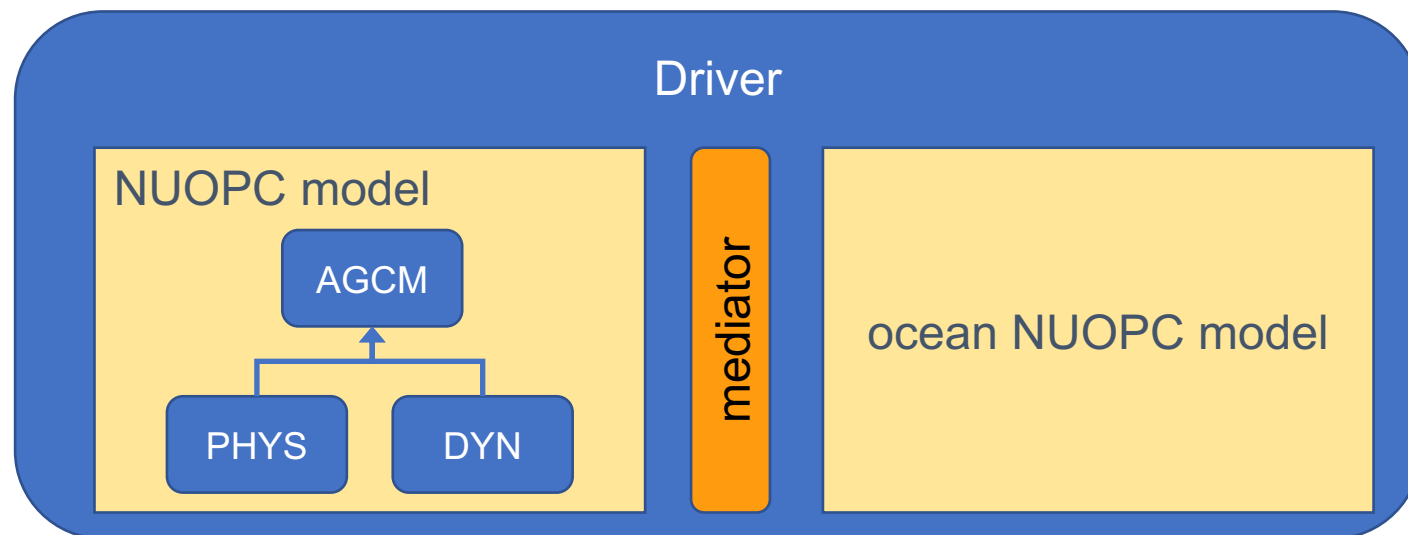
Backward (non) Compatibility

- ➢ Backward compatibility would be ideal, but …
  - ➢ not when it prevents new capabilities

- ➢ Varying levels of compliance
  - ➢ Staged approach to porting client grid comps

- ➢ **Level 0**: works with existing configurations (grid monoculture)
  - ➢ Parents cannot directly access states of children

- ➢ **Level 1**: can connect to components with different grids
  - ➢ Components cannot modify imports

- ➢ **Level 2**: can connect to components running on different devices
  - ➢ **2A**: must specify if run phase runs on device
  - ➢ **2B**: must specify which imports/exports are consumed/produced in each run phase

Current Status

➢ Expect to release in ~~2021~~, ~~2022~~, 2023

➢ Core framework is complete with extensive unit tests

➢ Partial implementation of various subclasses and hooks

➢ In process of converting user-lever interface wrappers from MAPL 2 to MAPL 3

Questions/discussion